# Лабораторная работа № 2 Системы контроля версий

**Цель работы:**

Получить опыт практической работы с системой контроля версий на примере BitBucket и TortoiseGit.

# Введение

Контроль версий подразумевает под собой комплекс методов, направленных на систематизацию изменений, вносимых разработчиками в программный продукт в процессе его разработки и сопровождения, сохранение целостности системы после изменений, предотвращение нежелательных и непредсказуемых эффектов. Также использование систем контроля версий позволяет сделать процесс внесения изменений более формальным.

Система управления версиями гарантирует, что каждый автор всегда работает с самой последней версией файла, а также исключает возможность случайной перезаписи любым из авторов работы своих коллег.

BitBucket – это сервис для разработчиков программного обеспечения, предоставляющий хостинг для проектов.

Одним из основных сервисов, предоставляемых BitBucket, является контроль версий. Доступные системы контроля версий – Git и Mercurial. Также доступны сервисы просмотра исходного кода, баг-трекинга, ведения wiki проекта и другие.

# Создание проекта на BitBucket

1. Зайдите на сайт https://bitbucket.org. Авторизуйтесь под своим аккаунтом google, либо создайте новый. При регистрации необходимо выбрать «5 users» в пункте «Plan».
2. Для воздания частного репозитория необходимо нажать «Create» **→** «Create Repository» в верхнем правом углу на главной странице (рис. 1).

В открывшейся странице (рис. 2):

* введите название проекта в поле «Name».
* введите краткое описание в поле «Description»
* выберите тип репозитория «Git» в поле «Repository type»
* отметьте пункт «Issue tracking» в поле «Project Management»
* нажмите кнопку «Create Repository»

После создания репозитория будет доступно его меню (рис. 1, слева). Меню состоит из двух групп «Actions» и «Navigation». Пункт меню «Clone» в группе

«Actions» позволяет получить адрес репозитория для дальнейшей работы с ним.

Пункт меню «Source» в группе «Actions» предназначен для просмотра исходного кода, пункт меню «Commits» позволяет просмотреть историю изменений, пункт

«Issues» Предназначен для учета ошибок и задач. Кроме того, есть возможно скачать репозиторий с помощью пункта «Downloads».

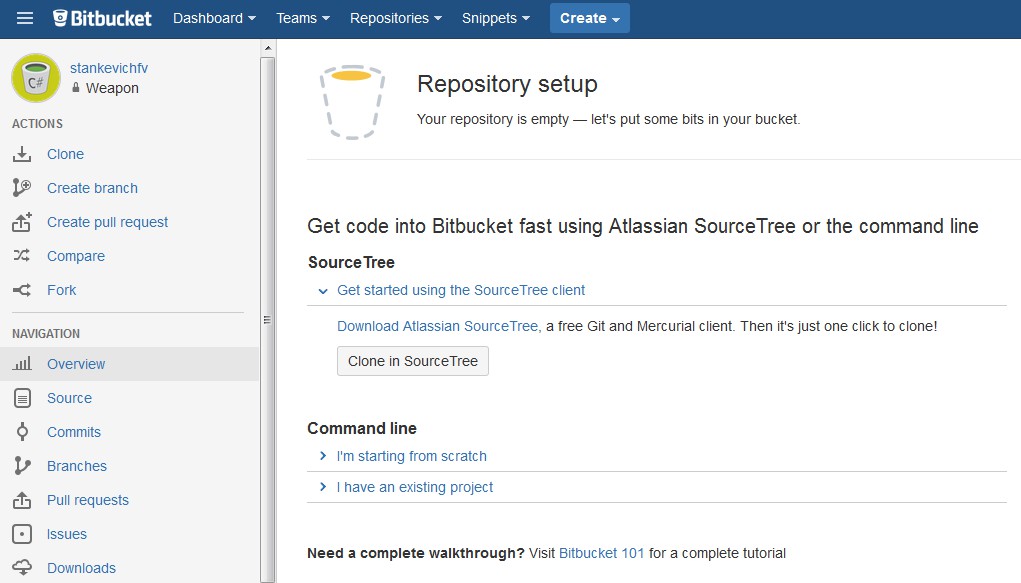


Рис. 1 Главная страница BitBucket

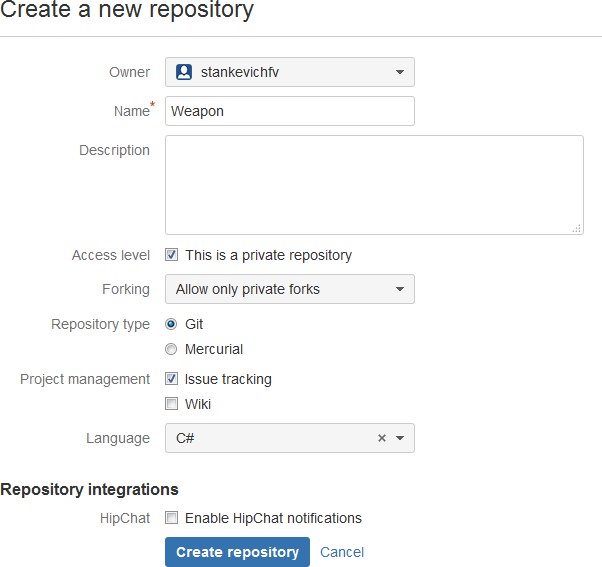


Рис. 2 Создание репозитория

# Система контроля версий Git

BitBucket – это централизированная система управления версиями. В настоящее время она используется многими сообществами разработчиков открытого программного обеспечения.

Git – распределённая система, где исходный код может храниться в разных хранилищах. Хранилище может располагаться на локальном диске или на сетевом сервере.

Для совместной работы над файлами в Git используется модель копирование – изменение – слияние. Кроме того, для файлов, не допускающих слияние (различные бинарные форматы файлов), можно использовать модель блокирование – изменение – разблокирование.

Для работы с Git будем использовать бесплатный клиент TortoiseGit, выполненный как расширение оболочки Windows.

# Начало работы с BitBucket

Для того чтобы скачать репозиторий проекта, необходимо сделать следующее:

1. Создайте папку, в которой будет храниться ваша локальная копия проекта (для удобства лучше хранить все репозитории в одной папке, например, C:\Git).
2. Нажимаем на созданную папку правой кнопкой мыши и из контекстного меню выбираем «Git Clone…» (рис. 3).

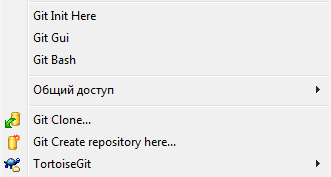


Рис. 3 Контекстное меню папки

1. В появившемся диалоговом окне (рис. 4) в поле «URL» введите полный URL репозитория Git (https://..., без команды *git clone*). (рис. 1, пункт

«Clone»)

1. Нажмите кнопку «OK», начнется скачивание репозитория с Git-сервера.
2. Затем система попросить ввести пароль от репозитория BitBucket.

**Модификация проекта**

C локальной рабочей копией можно работать как с обычной папкой: создавать, редактировать, удалять файлы и/или папки.

1. Добавим новые файлы в папку проекта (файл SampleProgram.cs).
2. Нажмите правой кнопкой мыши по папке проекта, из контекстного меню выберите «Git Commit -> “master” …».

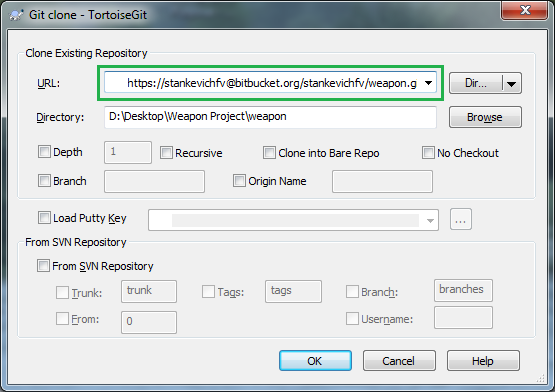


Рис. 4 Диалоговое окно «Git clone»

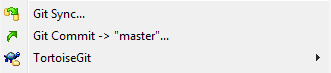


Рис. 5 Контекстное меню

1. При первом коммите TortoiseGit запросит ввести имя пользователя и электронную почту (рис. 6)

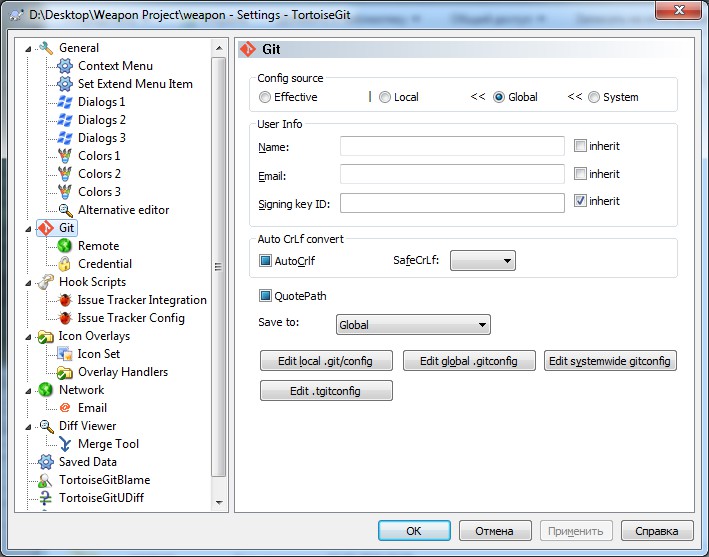


Рис. 6 Ввод имени и e-mail

1. В открывшемся окне (рис. 7) в секции «Message» введите краткое описание внесенных изменений (это рекомендуется делать всегда, особенно если не вы один работаете над проектом).
2. В секции «Changes made (double-click on file for diff)» вы увидите список всех изменённых, добавленных и удалённых вами файлов локальной рабочей копии. У изменённых файлов будут автоматически стоять флажки, у остальных — нет. Установите флажки у тех файлов, изменения которых должны быть загружены на git-сервер (есть флажок — файл обновляется, добавляется, удаляется; нет флажка — остаётся без изменений). Двойной щелчок по файлу запустит утилиту Tortoise Merge (предназначена для сравнения версий файлов), которая покажет последнюю версию файла из репозитория Git в левом окне и текущую рабочую копию в правом. Изменения будут выделены: удалённые строки зачёркнуты, добавленные выделены.

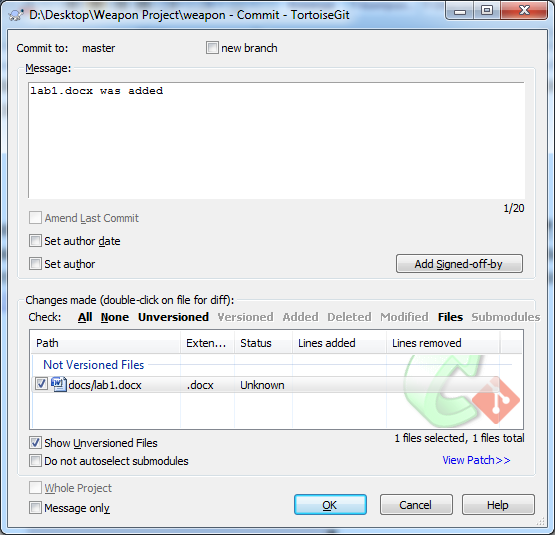


Рис. 7 Диалоговое окно «Commit».

1. После выделения нужных файлов или всех сразу, нажмите кнопку «OK». Произойдет фиксация внесенных вами изменений в локальном репозитории. Каждый раз при выполнении фиксации в репозитории создаётся новое состояние дерева файловой системы, называемое *ревизией*. Каждой ревизии назначается уникальный номер. Начальная ревизия не содержит ничего, кроме пустой корневой папки.
2. Для того чтобы отправить изменения на сервер необходимо нажать правой кнопкой мыши по папке проекта снова, из контекстного меню выбрать

«Git Sync» (рис. 8).

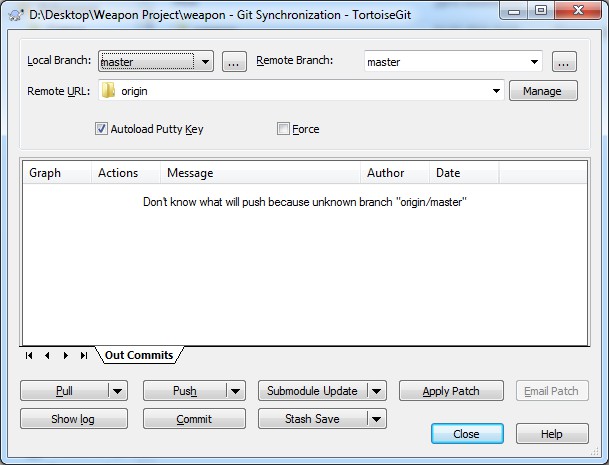


Рис. 8 Диалоговое окно «Git Sync».

1. В появившемся окне (рис. 8) нажмите «Push» и введите пароль. Кнопка

**Просмотр изменений**

Откройте добавленный ранее файл в любом текстовом редакторе и внесите в него любое изменение. Выполните «Commit» и «Push», как было описано выше.

Для просмотра изменений в проекте можно использовать инструменты сайта https://bitbucket.org (пункт меню «Commits» рис. 9 изменения выделяются цветом), так и возможности TortoiseGit (рис. 10 — 11).

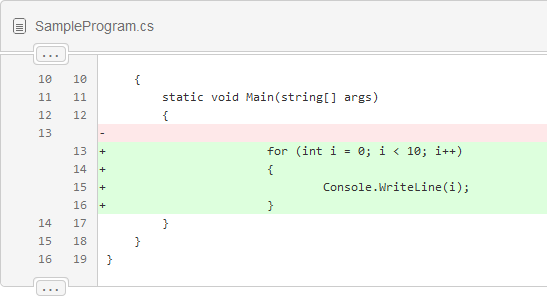


Рис. 9 Просмотр изменений

Вызовите контекстное меню любой папки или файла проекта (рис. 10). TortoiseGit позволяет просмотреть журнал изменений любого файла/каталога (пункт меню «Show log») или построить граф ревизий (пункт меню «Revision graph»). Граф ревизий полезен для больших проектов с несколькими отдельными ветвями разработки. Для отображения изменений в программном коде используется журнал изменений (рис. 11). Он показывает список всех фиксаций изменений в файле или папке, а также детальные сообщения разработчиков.

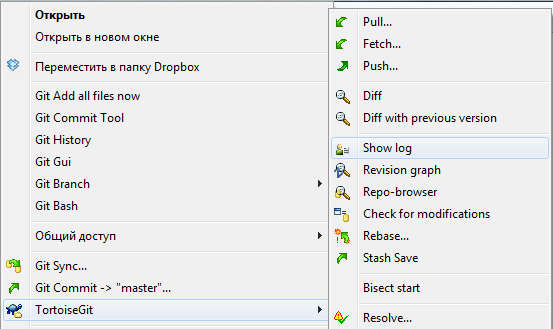


Рис. 10 Контекстное меню

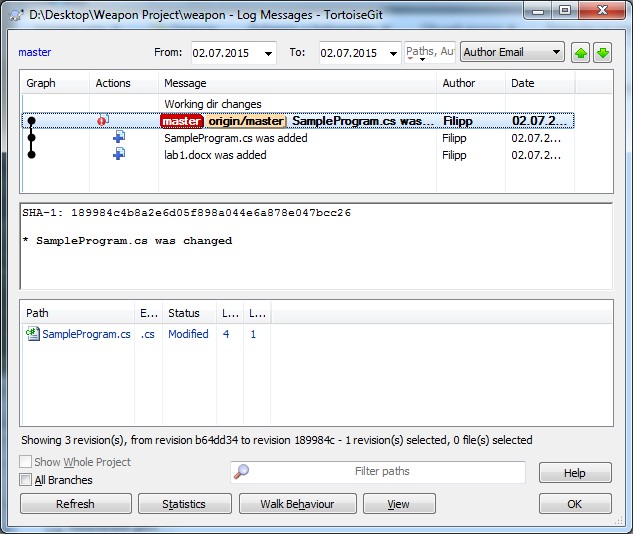


Рис. 11 Журнал изменений

Для просмотра отдельных изменений в коде щелкните два раза на изменении из списка. В результате откроется окно утилиты Tortoise Merge, отображающее внесенные изменения (рис. 11).

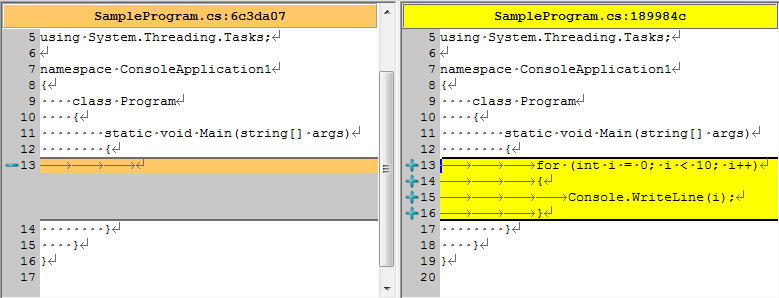


Рис. 12 Внесенные изменения

**Получение последней версии проекта**

Для получения изменений с сервера необходимо нажать кнопку «Pull» в диалоговом окне «Git Sync» (рис. 8) и дождаться, пока изменения загрузятся с сервера. Рекомендуется всегда перед «Commit» выполнять действие «Pull». Также «Pull» можно выполнить посредством контекстного меню папки проекта пункт “TortoiseGit” **→** “Pull” (рис. 10).

**Структура проекта в хранилище**

При разработке программного обеспечения основная задача делится на более мелкие, которые выполняются либо последовательно, либо параллельно относительно друг друга. При разработке выбирается основная ветвь разработки (в нее входят наиболее важные этапы, оказывающие влияние на весь проект), от нее отходят дочерние ветви, содержащие менее важные процессы. Таким образом, при разработке программных продуктов необходимо обеспечивать возможность одновременной разработки. Стратегически грамотное ветвление позволяет сохранять порядок и последовательность при работе с большим числом версий программного обеспечения.

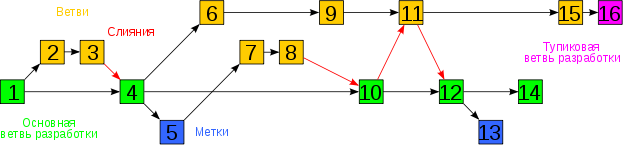


Рис. 13 Пример эволюции ветвей

Формально Git не накладывает каких-либо ограничений на файловую структуру проекта — она может быть какой угодно в рамках правил именования объектов файловой системы. Тем не менее, существуют рекомендации, призванные облегчить работу с ветвями и метками.

Основанная линия разработки проекта называется *trunk*. Дочерние же линии проекта именуются *branches.* Имя конкретной ветки (*branch*) можно выбрать в соответствии с разрабатываемым в ней функционалом, например *branch\_new\_gui*.

# Задание

1. Создайте новый проект на любой системе контроля версий.
2. Поместите в него текстовый документ, созданный в предыдущей лабораторной работе.
3. Работа в паре – первый участник вносит изменение в документ – второй просматривает изменения, вносит исправления. Просмотр истории изменений.

**Требования и рекомендации:**

Название проекта и все комментарии к коммитам должны быть осмыслены. Это необходимо для того, чтобы было удобнее понимать суть проекта и изменений, а также отслеживать их. Кроме того, это поможет человеку, увидевшему проект в первый раз, разобраться в нем.

# Контрольные вопросы

1. Что такое системы контроля версий?
2. Какие системы контроля версий вы знаете?
3. Какие существуют основные операции в системе контроля версий?

# Список литературы

* 1. Scott Chacon. Pro Git [Электронный ресурс]. — Режим доступа: https://git- scm.com/book/ru/v2, свободный.
  2. Mike McQuaid. Git in Practice, Manning, 2014.
  3. Bitbucket Documentation [Электронный ресурс]. — Режим доступа: https://confluence.atlassian.com/display/BITBUCKET/Bitbucket+Documentation+Home, свободный.